

HashiCorp Terraform Associate (003) Exam Guide

Preface:

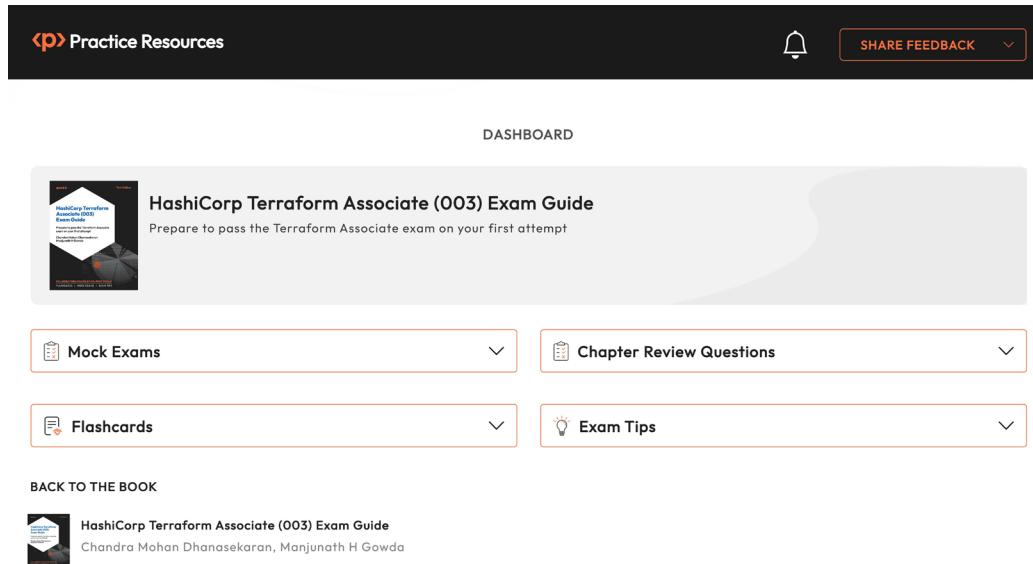


Figure 0.1 – Online exam-prep platform on a desktop device

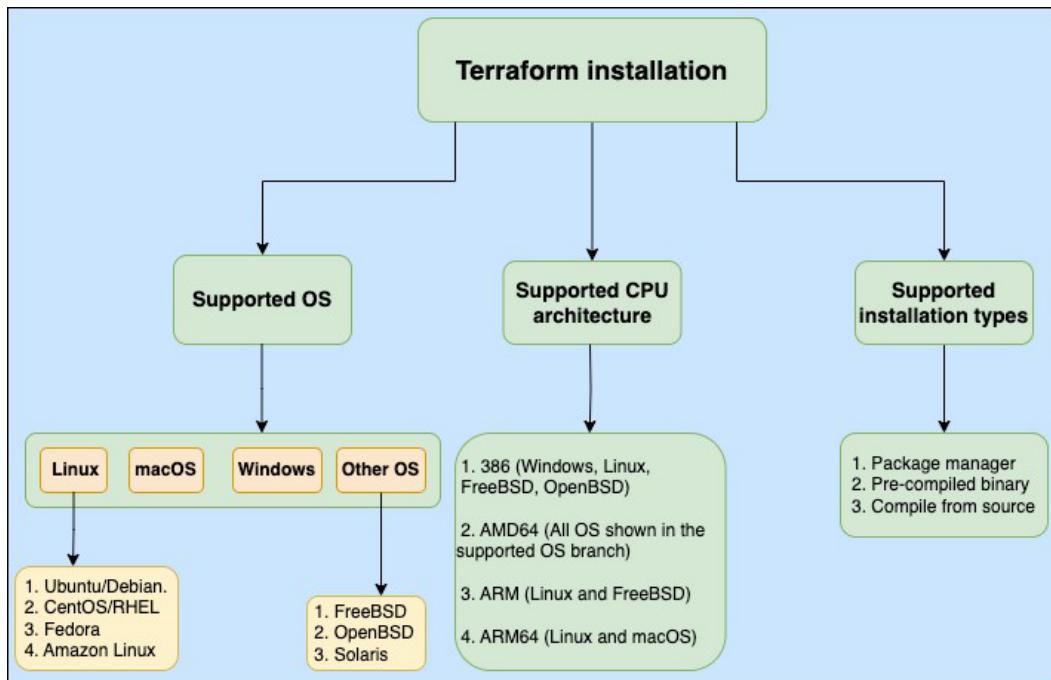


Figure 0.2: Terraform installation

```
~$brew install hashicorp/tap/terraform
=> Fetching hashicorp/tap/terraform
=> Downloading https://releases.hashicorp.com/terraform/1.7.3/terraform_1.7.3_darwin_arm64.zip
Already downloaded: /Users/manjunath/Library/Caches/Homebrew/downloads/a0ca203fddc90ae140c9d76ebdf8991f960ab3c01a8baf1df77626550e389b7d6--terraform_1.7.3_darwin_arm64.zip
=> Installing terraform from hashicorp/tap
=> Downloading https://formulae.brew.sh/api/formula.jws.json
#####
=> Downloading https://formulae.brew.sh/api/cask.jws.json
#####
Warning: Cask homebrew/cask/terraform was renamed to homebrew/core/terraform.
D /opt/homebrew/Cellar/terraform/1.7.3: 3 files, 88.6MB, built in 4 seconds
=> Running `brew cleanup terraform`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
~$
```

Figure 0.3: Terraform installation using Homebrew

S

```
~$terraform -v
Terraform v1.7.3
on darwin_arm64
+ provider registry.terraform.io/hashicorp/aws v4.64.0
~$
```

Figure 0.4: Terraform installation validation

Binary download

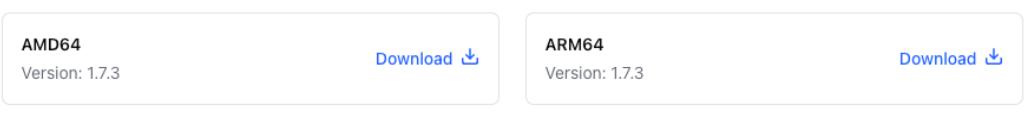


Figure 0.5: Pre-compiled binary for macOS

```
~/Downloads $ls
terraform_1.7.3_darwin_arm64.zip
~/Downloads $
~/Downloads $unzip terraform_1.7.3_darwin_arm64.zip
Archive: terraform_1.7.3_darwin_arm64.zip
  inflating: terraform
~/Downloads $
~/Downloads $ls --color=auto
terraform               terraform_1.7.3_darwin_arm64.zip
~/Downloads $
```

Figure 0.6: Unzipping the Terraform ZIP file

```
~/Downloads $terraform -v
Terraform v1.7.3
on darwin_arm64
```

Figure 0.7: Pre-compiled binary installation validation

```

PS C:\Users\Administrator> choco install terraform
Chocolatey v2.2.2
Installing the following packages:
terraform
By installing, you accept licenses for the packages.
Progress: Downloading terraform 1.7.3... 100%

terraform v1.7.3 [Approved]
terraform package files install completed. Performing other installation steps.
The package terraform wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint): Y

Removing old terraform plugins
Downloading terraform 64 bit
  from 'https://releases.hashicorp.com/terraform/1.7.3/terraform_1.7.3_windows_amd64.zip'
Progress: 100% - Completed download of C:\Users\Administrator\AppData\Local\Temp\chocolatey\terraform\1.7.3\terraform_1.7.3_windows_amd64.zip (25.05 MB).
Download of terraform_1.7.3_windows_amd64.zip (25.05 MB) completed.
Hashes match.
Extracting C:\Users\Administrator\AppData\Local\Temp\chocolatey\terraform\1.7.3\terraform_1.7.3_windows_amd64.zip to C:\ProgramData\chocolatey\lib\terraform\tools...
C:\ProgramData\chocolatey\lib\terraform\tools
ShimGen has successfully created a shim for terraform.exe

```

Figure 0.9: Terraform installation via Chocolatey

1.

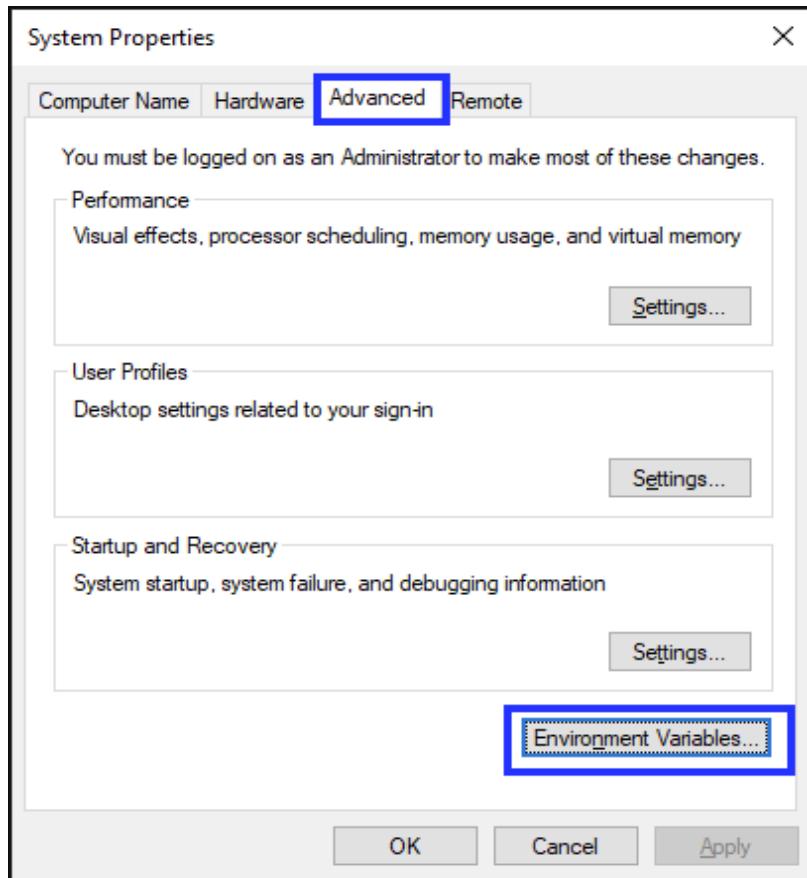


Figure 0.10: System properties

1.

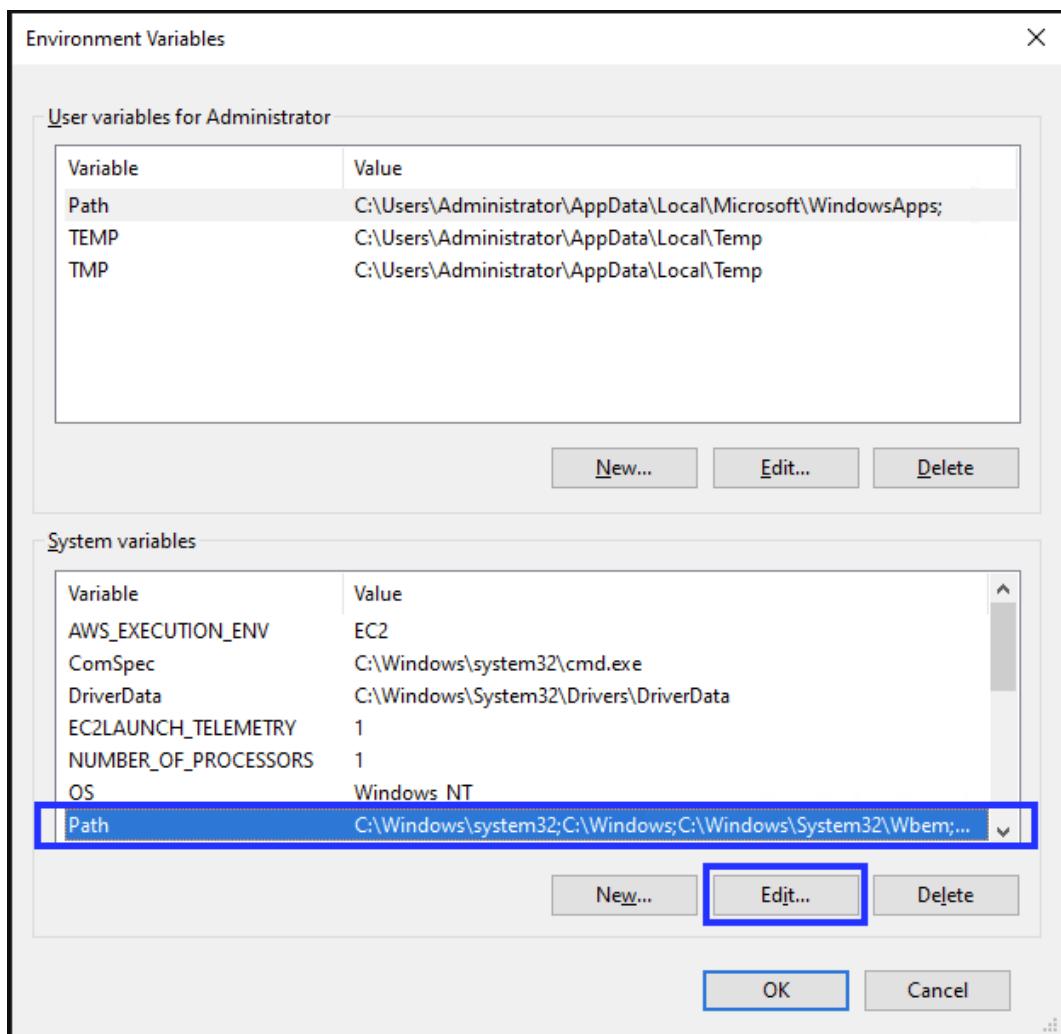


Figure 0.11: Modifying the PATH environment variable

2.

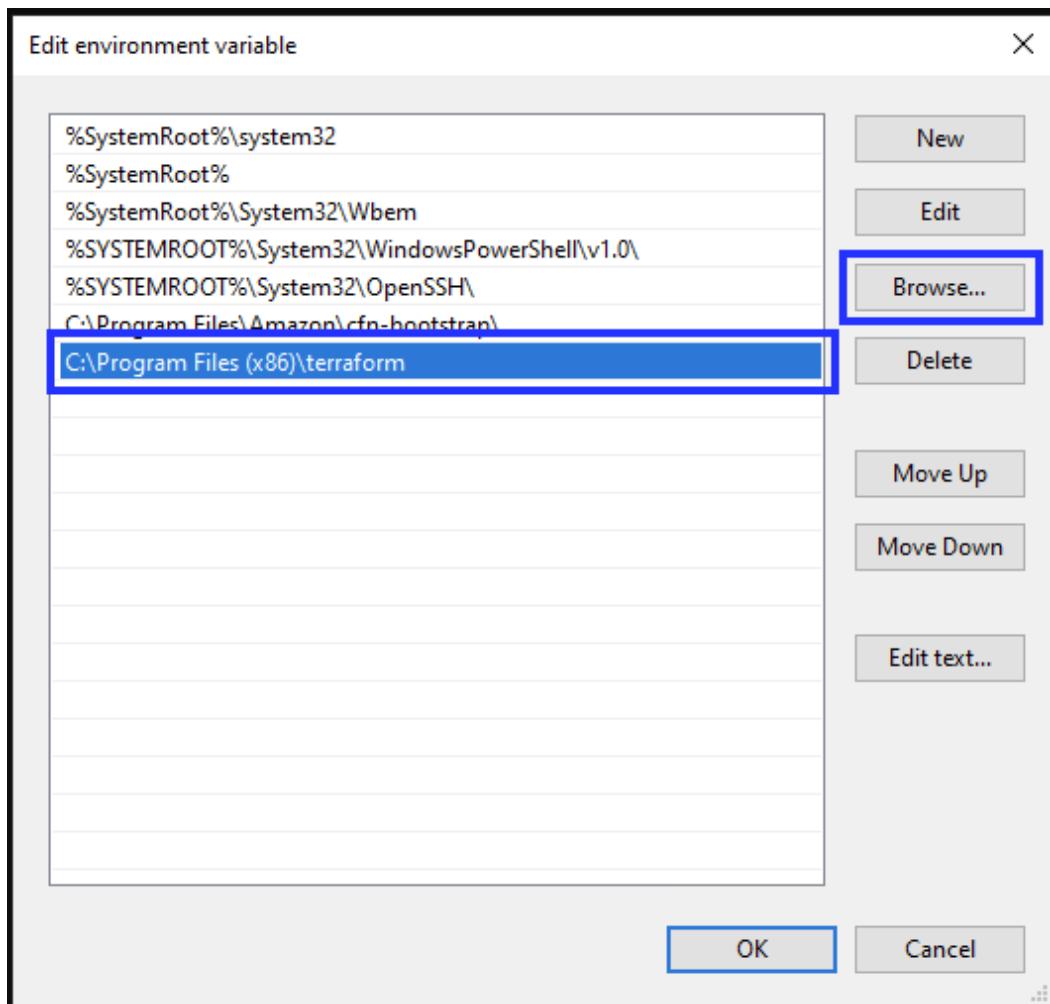


Figure 0.12: Adding the Terraform location to the PATH environment variable

```
~ $aws configure
AWS Access Key ID [None]: AKIAYC3
AWS Secret Access Key [None]: EdMKRvx40S5[REDACTED]4VJo5NzU05gA
Default region name [None]: ap-south-1
Default output format [None]: json
~ $
```

Figure 0.13: Configuring the AWS CLI

```

● /Users/packt/Desktop $terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.37.0...
- Installed hashicorp/aws v5.37.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
● /Users/packt/Desktop $terraform plan

No changes. Your infrastructure matches the configuration.

```

Figure 0.14: Configuring AWS provider

```

# module.vpc.aws_vpc.this[0] will be created
+ resource "aws_vpc" "this" {
    + arn                               = (known after apply)
    + cidr_block                         = "10.0.0.0/16"
    + default_network_acl_id            = (known after apply)
    + default_route_table_id           = (known after apply)
    + default_security_group_id        = (known after apply)
    + dhcp_options_id                  = (known after apply)
    + enable_dns_hostnames             = true
    + enable_dns_support               = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                                = (known after apply)
    + instance_tenancy                 = "default"
    + ipv6_association_id              = (known after apply)
    + ipv6_cidr_block                  = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id              = (known after apply)
    + owner_id                          = (known after apply)
    + tags
        + "Name" = ""
    }
    + tags_all                         = (known after apply)
}

Plan: 4 to add, 0 to change, 0 to destroy.

```

Figure 0.15: Terraform plan for VPC creation in AWS

Chapter 1: Introduction to Infrastructure as Code (IaC) and Concepts

The screenshot shows a dark-themed dashboard interface. At the top left is a 'Practice Resources' section with a bell icon and a 'SHARE FEEDBACK' button. Below this is a 'DASHBOARD' section featuring a card for the 'HashiCorp Terraform Associate (003) Exam Guide'. The card includes a small thumbnail image, the title, and a subtitle: 'Prepare to pass the Terraform Associate exam on your first attempt'. Below the card are four expandable sections: 'Mock Exams', 'Chapter Review Questions', 'Flashcards', and 'Exam Tips'. At the bottom left is a link to 'BACK TO THE BOOK' with a thumbnail of the book cover and its title.

Figure 1.1 – Dashboard interface

Infrastructure as Code (IaC)

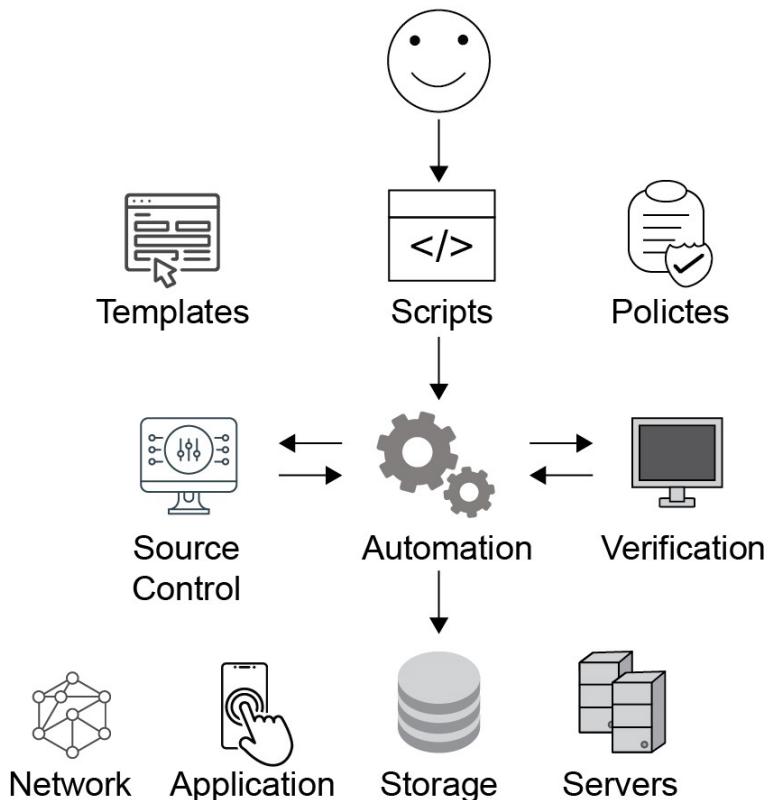


Figure 1.2 – IaC workflow

The screenshot shows the AWS DynamoDB 'Tables' page. At the top, there is a search bar labeled 'Find tables by table name' and filters for 'Any tag key' and 'Any tag value'. Below the header, there is a table with columns: Name, Status, Partition key, Sort key, and Indexes. A single row is visible, representing a table named 'DynamoDB-Test-myDynamoDBTable-QEC9O7JOSHL6' which is 'Active' with a partition key 'CustomerID (S)' and 0 sort keys.

Name	Status	Partition key	Sort key	Indexes
DynamoDB-Test-myDynamoDBTable-QEC9O7JOSHL6	Active	CustomerID (S)	-	0

Figure 1.3 – Table in the console

The screenshot shows the output of the Terraform 'init' command. It starts with 'Initializing the backend...', followed by 'Initializing provider plugins...' which lists the installation of 'hashicorp/aws' version v5.15.0. Then it states that a lock file '.terraform.lock.hcl' has been created to record provider selections. Finally, it confirms that Terraform has been successfully initialized and provides instructions for working with the infrastructure.

```
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.14"...
- Installing hashicorp/aws v5.15.0...
- Installed hashicorp/aws v5.15.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Figure 1.4: The output will look like this

The screenshot shows the AWS S3 'Buckets' page. At the top, there is a search bar labeled 'Q test' and a filter for 'AWS Region' set to 'US East (N. Virginia) us-east-1'. Below the header, there is a table with columns: Name, AWS Region, Access, and Creation date. A single row is visible, representing a bucket named 'test-s3-bucket-kquiyr' located in 'us-east-1' with access set to 'Bucket and objects not public' and created on 'September 1, 2023, 05:54:07 (UTC+05:30)'.

Name	AWS Region	Access	Creation date
test-s3-bucket-kquiyr	US East (N. Virginia) us-east-1	Bucket and objects not public	September 1, 2023, 05:54:07 (UTC+05:30)

Figure 1.5 – Bucket in the console

test-s3-bucket-kquiyrt [Info](#)

Objects | **Properties** | Permissions | Metrics | Management | Access Points

Bucket overview

AWS Region US East (N. Virginia) us-east-1	Amazon Resource Name (ARN) arn:aws:s3:::test-s3-bucket-kquiyrt	Creation date September 1, 2023, 05:54:07 (UTC+05:30)
---	---	--

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

[Edit](#)

Bucket Versioning
Enabled
Multi-factor authentication (MFA) delete

Figure 1.6 – S3 bucket with versioning option enabled

◀p Practice Resources

DASHBOARD > CHAPTER 1

Introduction to Infrastructure as Code (IaC) and Concepts

Summary

In this chapter, the IaC concept was discussed in detail, and some popular IaC tools were covered to help you understand the differences between them. You then read through various use cases demonstrating the implementation of the IaC approach, and these were followed up with the benefits of using them.

After having done the practical exercises using CloudFormation and Terraform tools, you now also have a better idea about the actual working of these tools and how they differ from each other.

Chapter Review Questions

The HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda

Select Quiz

Quiz 1 [SHOW QUIZ DETAILS](#) [START](#)

Figure 1.8 – Chapter Review Questions for Chapter 1

Chapter 2: Why Do We Need Terraform?

The screenshot shows a dark-themed web application interface. At the top left is a logo with the letters 'P' and 'R'. To its right is the text 'Practice Resources'. On the far right are a bell icon and a 'SHARE FEEDBACK' button with a dropdown arrow. Below this header, a breadcrumb navigation shows 'DASHBOARD > CHAPTER 2'. The main content area has a light gray background. On the left, a white box contains the title 'Why Do We Need Terraform?' and a 'Summary' section. The summary text discusses the challenges of manual provisioning, what IaC is, and Terraform's importance in the IaC landscape. It also mentions the change in license and the readiness to explore Terraform basics. On the right, a dark box titled 'Chapter Review Questions' displays the title 'Chapter Review Questions' and the subtitle 'The HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda'. It features a 'Select Quiz' section with a 'Quiz 1' entry, a 'SHOW QUIZ DETAILS' link, and an orange 'START' button.

Figure 2.2 – Chapter Review Questions for Chapter 2

Chapter 3: Basics of Terraform and Core Workflow



Figure 3.1 – A high-level overview of Terraform providers

```
C:\Users>terraform -version
Terraform v1.5.7
on windows_amd64
```

Figure 3.2 – Terraform version command output

```
C:\Users>aws --version
aws-cli/2.15.16 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
```

Figure 3.3 – AWS CLI version command output

```
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.19.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Figure 3.4 – terraform init command output

```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# aws_iam_access_key.user_access_keys will be created
+ resource "aws_iam_access_key" "user_access_keys" {
    + create_date          = (known after apply)
    + encrypted_secret     = (known after apply)
    + encrypted_ses_smtp_password_v4 = (known after apply)
    + id                   = (known after apply)
    + key_fingerprint      = (known after apply)
    + secret               = (sensitive value)
    + ses_smtp_password_v4 = (sensitive value)
    + status               = "Active"
    + user                 = "test-aws-user"
}

# aws_iam_user.test_user will be created
+ resource "aws_iam_user" "test_user" {
    + arn                  = (known after apply)
    + force_destroy        = false
    + id                  = (known after apply)
    + name                = "test-aws-user"
    + path                = "/test/"
    + tags                = {
        + "createdby" = "terraform"
    }
    + tags_all            = {
        + "createdby" = "terraform"
    }
    + unique_id           = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: █

```

Figure 3.5 – terraform apply console output

```

aws_iam_user.test_user: Creating...
aws_iam_user.test_user: Creation complete after 1s [id=test-aws-user]
aws_iam_access_key.user_access_keys: Creating...
aws_iam_access_key.user_access_keys: Creation complete after 0s [id=██████████]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

```

Figure 3.6 – resource creation from terraform apply



Figure 3.7 – Official provider: aws



Partner by:oracle

Infrastructure (IaaS)

Figure 3.8 – Partner provider: oci

A screenshot of a web-based learning platform. At the top, there's a navigation bar with a 'Practice Resources' icon, the text 'Practice Resources', a bell icon, and a 'SHARE FEEDBACK' button. Below the navigation, the text 'DASHBOARD > CHAPTER 3' is visible. The main content area has a dark background. On the left, under 'Basics of Terraform and Core Workflow' and 'Summary', there's a text block about Terraform basics and a note about the next chapter. On the right, under 'Chapter Review Questions', it says 'The HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda'. It features a 'Select Quiz' section with 'Quiz 1' and a 'START' button, along with a 'SHOW QUIZ DETAILS' link.

Figure 3.11 – Chapter Review Questions for Chapter 3

Chapter 4: Terraform Commands and State Management

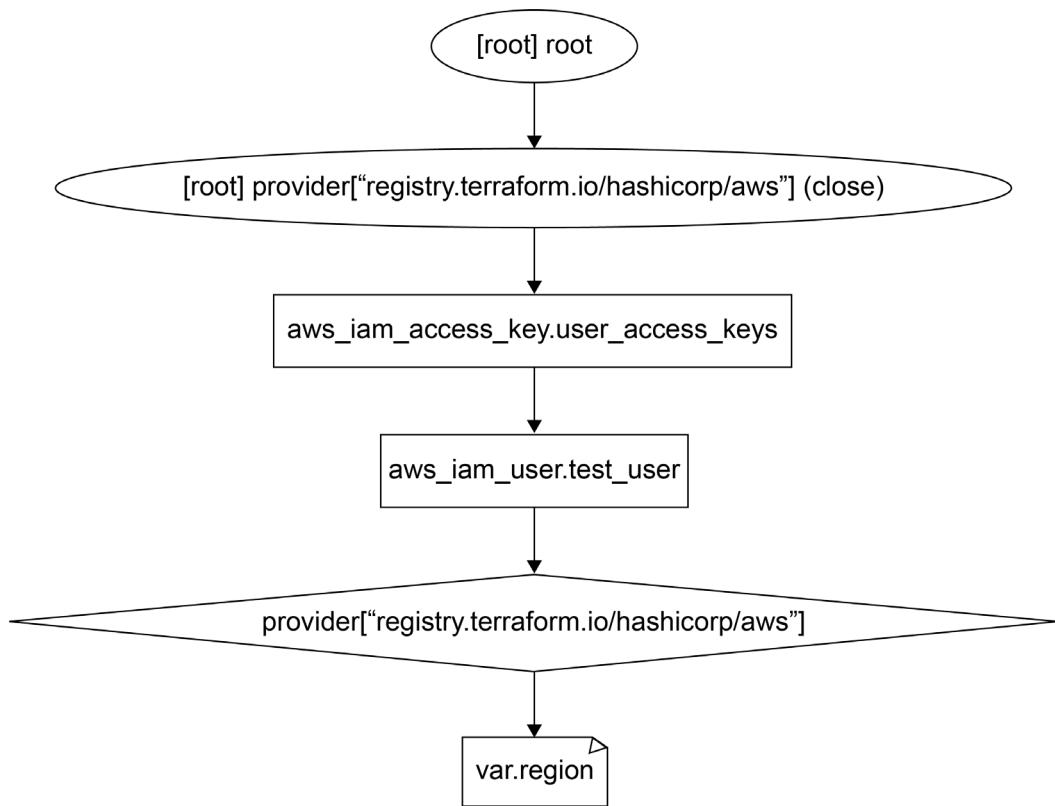


Figure 4.1: The graph command output

In this chapter, you read about the core Terraform workflow commands. This was followed by explanations of special-purpose commands such as console, graph, and import, used as and when the need arises. The terraform state command was then discussed in detail with the subcommands. You also looked at how each command helps the developer interact with and manage the state file beyond infrastructure provisioning.

Now, you can proceed with the next chapter, on how to use the commands with Terraform CLI with full confidence. This will help in increasing productivity and the seamless transition to advanced concepts such as Terraform modules, Terraform Cloud, and so on.

Chapter Review Questions

The HashiCorp Terraform Associate (003) Exam Guide
by Chandra Mohan Dhanasekaran, Manjunath H Gowda

Select Quiz

Quiz 1 START SHOW QUIZ DETAILS

Figure 4.3 – Chapter Review Questions for Chapter 4

Chapter 5: Terraform Modules

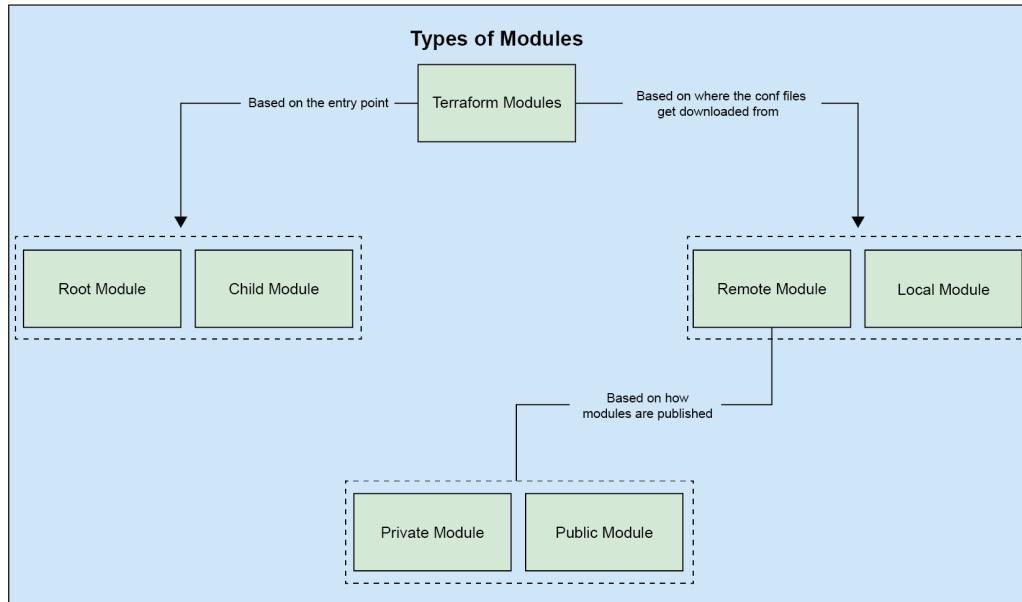


Figure 5.1: Types of modules

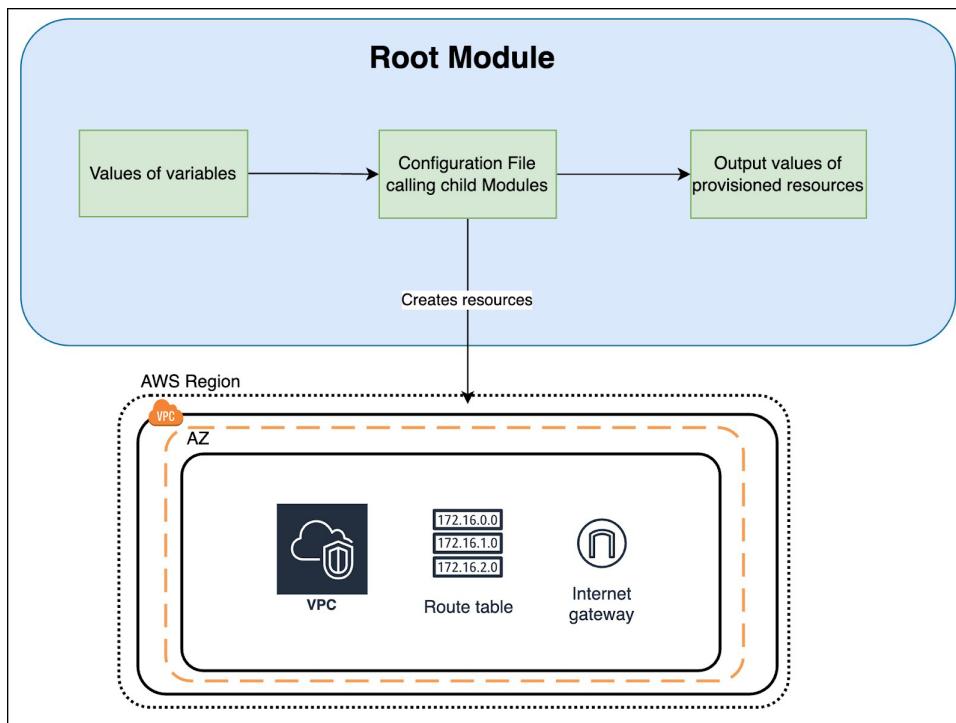


Figure 5.2: Root module

The screenshot shows a code editor interface with the following details:

- EXPLORER** pane on the left:
 - APP** folder:
 - main.tf
 - outputs.tf
 - README.md
 - terraform.tfvars
 - variables.tf** (highlighted)
- TERMINAL** pane on the right:


```
• ~/terraform-code/app $tree
```

```
- README.md
- main.tf
- outputs.tf
- terraform.tfvars
- variables.tf
```

1 directory, 5 files

Figure 5.3: Root module file structure example

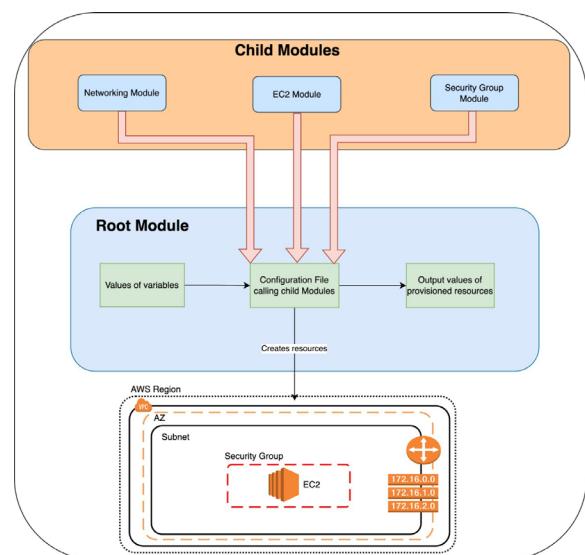


Figure 5.4: Child module called by root module

The screenshot shows a code editor interface with the following details:

- EXPLORER** pane on the left:
 - APP** folder:
 - modules** folder:
 - ec2** folder:
 - main.tf
 - outputs.tf
 - README.md
 - variables.tf
 - networking** folder:
 - main.tf
 - outputs.tf
 - README.md
 - variables.tf
 - security-group** folder:
 - main.tf
 - outputs.tf
 - README.md
 - variables.tf
 - OUTLINE** pane at the bottom
 - TERMINAL** pane on the right:


```
• ~/terraform-code/app $tree
```

```
- README.md
- main.tf
- modules
  - ec2
    - README.md
    - main.tf
    - outputs.tf
    - variables.tf
  - networking
    - README.md
    - main.tf
    - outputs.tf
    - variables.tf
  - security-group
    - README.md
    - main.tf
    - outputs.tf
    - variables.tf
- outputs.tf
- providers.tf
- variables.tf
```

Figure 5.5: Child module and root module file structure example

```

### CREATING VPC, SUBNET, ROUTE TABLE, IGW ####
module "vpc" {
  source      = "./modules/networking"
  vpc_cidr_block = "10.51.0.0/16"
  vpc_name      = "child-module-vpc"
  subnet_cidr_block = "10.51.1.0/24"
  az            = "ap-south-1a"
  igw_name      = "child-module-igw"
}

### CREATING SECURITY GROUP AND RULES ####
module "security_group" {
  source      = "./modules/security-group"
  vpc_id      = module.vpc.vpc_id
  security_group_name = "child-module-sg"
  allowed_ip    = "0.0.0.0/0"
  from_port     = "22"
  to_port       = "22"
}

```

source path is within
the local filesystem

Figure 5.6: Example code for local module implementation

```

• ~/terraform-code/app $ terraform init

Initializing the backend...
Initializing modules...
- ec2_instance in modules/ec2
- security_group in modules/security-group Local child modules
- vpc in modules/networking getting initialized

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.34.0"...
- Installing hashicorp/aws v5.34.0...
- Installed hashicorp/aws v5.34.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

```

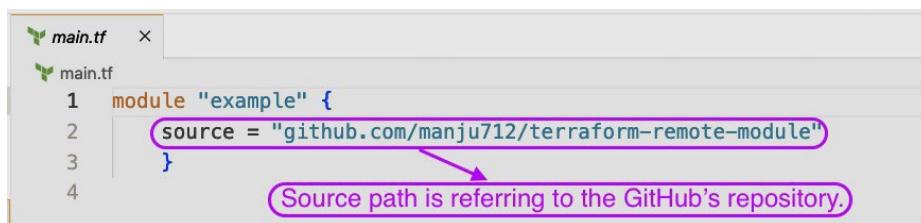
Figure 5.7: Local child modules getting initialized

```

module "ec2_instance" {
  source          = "terraform-aws-modules/ec2-instance/aws"
  version        = "5.6.0"
  name           = "public-module-ec2"
  ami             = data.aws_ami.root_module_ami.id
  instance_type  = "t3.micro"
  subnet_id      = module.vpc.public_subnets[0]
  vpc_security_group_ids = [module.security_group.security_group_id]
  key_name        = aws_key_pair.root_module_kp.id
  tags = {
    Name = "public-module-ec2"
  }
}

```

Figure 5.8: Example code for a public (and remote) module (Terraform Registry)



```

main.tf
main.tf
1 module "example" {
2   source = "github.com/manju712/terraform-remote-module"
3 }
4

```

A screenshot of a code editor showing a file named 'main.tf'. The code defines a module named 'example' with a single argument 'source' set to 'github.com/manju712/terraform-remote-module'. A callout box highlights the 'source' line with the text 'Source path is referring to the GitHub's repository.'

Figure 5.9: Example code for public (and remote) module (GitHub)

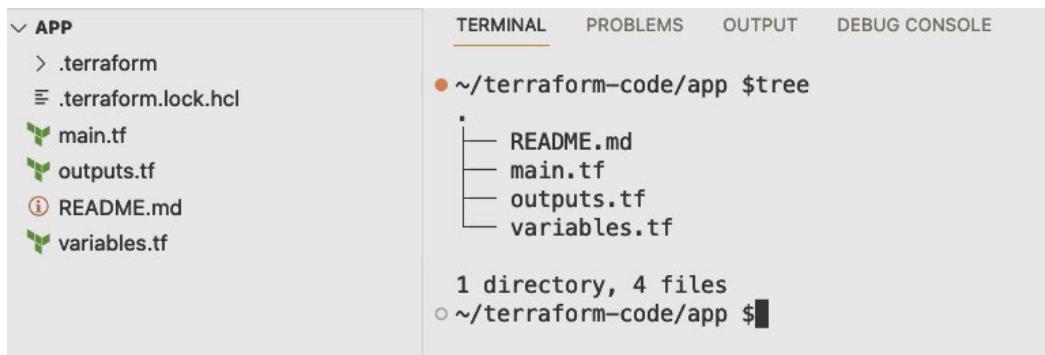


Figure 5.10: Module structure with minimal files

```

### GENERIC SYNTAX OF MODULE ####
module "local-name-for-the-module" {
  source    = "Local/Remote location of the child module"
  version   = "Version number of the module to be downloaded from the above source"
  VARIABLE1 = Value1
  VARIABLE2 = Value2
}

### EXAMPLE OF A MODULE ####
module "key_pair" {
  source  = "terraform-aws-modules/key-pair/aws"
  version = "2.0.2"

  key_name        = "my-keypair"
  create_private_key = true
}

```

Figure 5.11: Syntax and example of a module block

S

The screenshot shows a dark-themed web application interface. At the top, there's a navigation bar with the 'Practice Resources' logo, a bell icon for notifications, and a 'SHARE FEEDBACK' button. Below the navigation, the page title is 'Terraform Modules'. A 'Summary' section contains text about the chapter's content, mentioning learning about Terraform modules, their types, and how to use them. To the right, a 'Chapter Review Questions' section is displayed, featuring a sub-section for 'The HashiCorp Terraform Associate (003) Exam Guide' by Chandra Mohan Dhanasekaran, Manjunath H Gowda. It includes a 'Select Quiz' button, a 'Quiz 1' link, and a 'START' button.

Figure 5.13 – Chapter Review Questions for Chapter 5

Chapter 6: Terraform Backends and Resource Management

The screenshot shows a dark-themed user interface for 'Practice Resources'. At the top left is the logo 'cp Practice Resources'. On the right are a bell icon and a 'SHARE FEEDBACK' button with a dropdown arrow. Below the header, the navigation path is 'DASHBOARD > CHAPTER 6'. The main content area has a title 'Terraform Backends and Resource Management' and a 'Summary' section. The summary text discusses backend configuration syntax, valid backends, resource addressing, implicit and explicit dependency, and various expression types like splat and for expressions. To the right is a 'Chapter Review Questions' sidebar. It includes the title 'Chapter Review Questions', a subtitle 'The HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda', a 'Select Quiz' button, and a 'Quiz 1' section with a 'SHOW QUIZ DETAILS' link and an orange 'START' button.

Figure 6.4 – Chapter Review Questions for Chapter 6

Chapter 7: Debugging and Troubleshooting Terraform

The screenshot shows a dark-themed web interface for a chapter review. At the top, there's a header bar with a bell icon and a 'SHARE FEEDBACK' button. Below the header, a navigation bar shows 'DASHBOARD > CHAPTER 7'. The main content area has a title 'Debugging and Troubleshooting Terraform' and a 'Summary' section. The summary text discusses learned areas, enabling logging, and tips for real-world management. To the right, a 'Chapter Review Questions' sidebar lists the HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda. It features a 'Select Quiz' section with 'Quiz 1' and a 'START' button.

Practice Resources

DASHBOARD > CHAPTER 7

Debugging and Troubleshooting Terraform

Summary

In this chapter, you learned about the key areas in which Terraform issues arise and how to handle those issues. The areas or the issues highlighted here are in no way exhaustive but should give you a good understanding of the frequent issues in Terraform.

You also learned how to enable logging, which is the most important step in troubleshooting issues or raising a bug report with the providers or with HashiCorp.

In the end, you saw some tips from real-world Terraform management. In *Chapter 8, Terraform Functions*, you will read about the use of Terraform functions that the Terraform language supports.

Chapter Review Questions

The HashiCorp Terraform Associate (003) Exam Guide
by Chandra Mohan Dhanasekaran, Manjunath H Gowda

Select Quiz

Quiz 1 [SHOW QUIZ DETAILS](#) [START](#)

Figure 7.2 – Chapter Review Questions for Chapter 7

Chapter 8: Terraform Functions

```
main.tf > ...
1   variable "test-sensitive" {
2     type      = string
3     sensitive = true
4     default   = "test-string-1"
5   }
6
7   variable "test-nonsensitive" {
8     type      = string
9     sensitive = false
10    default   = "test-string-2"
11  }
12
13  output "outval1" {
14    value      = var.test-sensitive
15    sensitive = true
16  }
17
18  output "outval2" {
19    value = var.test-nonsensitive
20  }
21
```

Figure 8.1 – main.tf with sensitive and nonsensitive variables

```
rm-functions-new\type-conversion-functions> terraform plan
Changes to Outputs:
+ outval1 = (sensitive value)
+ outval2 = "test-string-2"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.
```

Figure 8.2 – The terraform plan terminal output

The screenshot shows the Practice Resources interface. At the top, there's a navigation bar with the 'Practice Resources' logo, a bell icon for notifications, and a 'SHARE FEEDBACK' button. Below the navigation bar, the path 'DASHBOARD > CHAPTER 8' is visible. The main content area is titled 'Terraform Functions' and has a 'Summary' section. The summary text states: 'In this chapter, you looked at the different types of functions, starting with basic numeric and string functions, followed by collection functions to work with the complex types. Then, the type conversion functions were covered with file-based functions with encoding options and a few special utility functions.' It also notes: 'From the exam perspective, these functions may not be directly asked about in the questions, but this knowledge will be helpful in other scenario-based questions.' To the right of the summary is a dark-themed box titled 'Chapter Review Questions'. It contains the text: 'The HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda'. Below this, there's a 'Select Quiz' section with a 'Quiz 1' entry and a 'START' button. There's also a 'SHOW QUIZ DETAILS' link.

Figure 8.4 – Chapter Review Questions for Chapter 8

Chapter 9: Understanding HCP Terraform's Capabilities

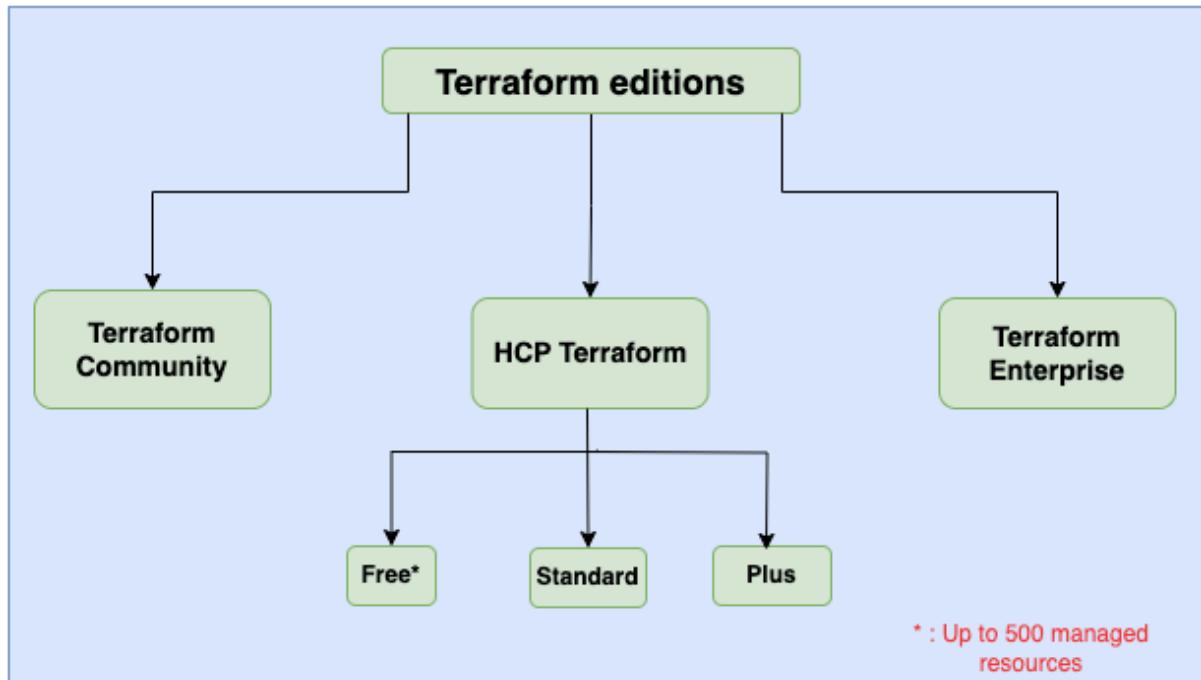


Figure 9.1: Terraform editions

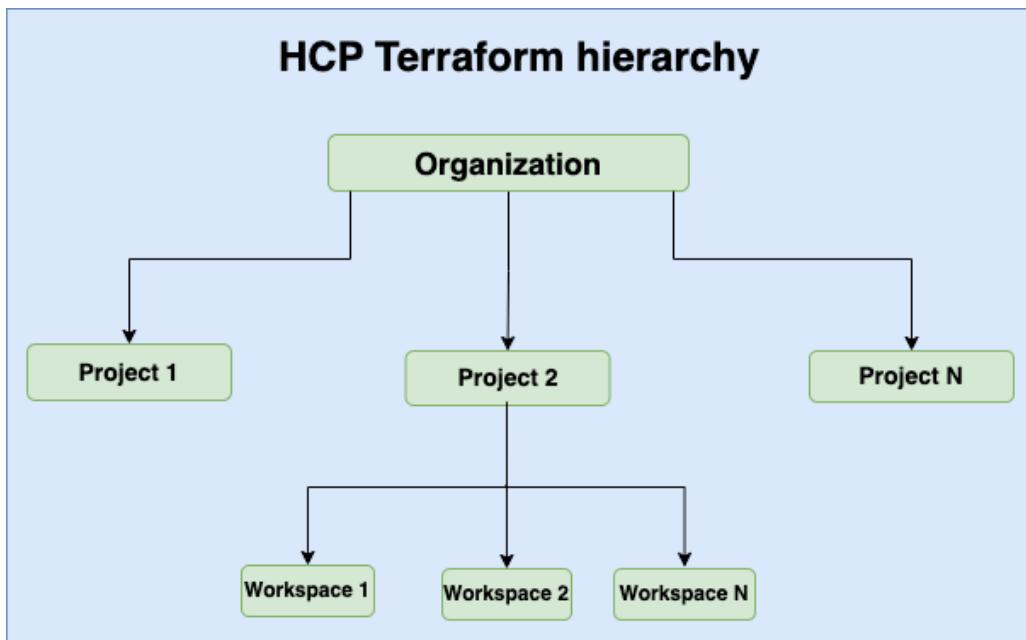


Figure 9.2: HCP Terraform concepts hierarchy

Create an account

Have an account? [Sign in](#)

 Continue with HCP account

OR

Username

packt-terraform-book

Email

packt-terraform@gmail.com



Password



I agree to the Terms of Use.

I acknowledge the Privacy Policy.

Please review the [Terms of Use](#) and [Privacy Policy](#).

[Create account](#)

Figure 9.3: HCP Terraform account sign up



Create Your Account

Sign up for HashiCorp to continue

Email address

packt-terraform@gmail.com

More options

Continue

Already have an account? [Log in](#)

OR

Continue with GitHub

Service agreements

Terms of Service Required

I accept the Terms of Service

Privacy Policy Required

I accept the Privacy Policy

Marketing and newsletters

I would like to be updated via email about HashiCorp products, features, events, announcements, and education materials.

Continue

Figure 9.4: HCP sign-up screens

```
~ $terraform login
Terraform will request an API token for app.terraform.io using your browser.

If login is successful, Terraform will store the token in plain text in
the following file for use by subsequent commands:
  /Users/packt/.terraform.d/credentials.tfrc.json

Do you want to proceed?
  Only 'yes' will be accepted to confirm.

Enter a value:
```

Figure 9.5: HCP Terraform login via CLI

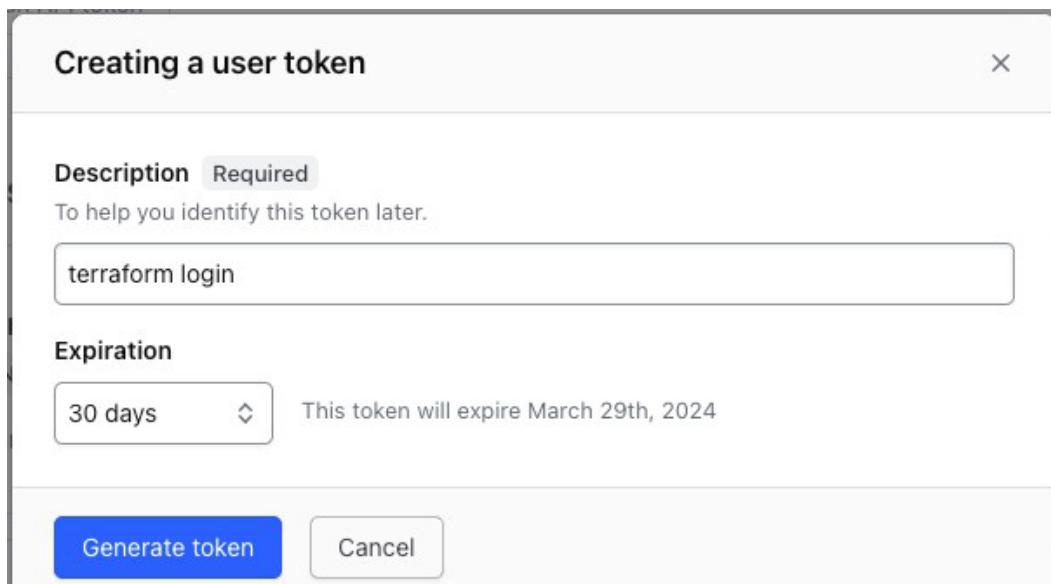


Figure 9.6: API token creation in HCP Terraform

```
-
Terraform must now open a web browser to the tokens page for app.terraform.io.
If a browser does not open this automatically, open the following URL to proceed
:
  https://app.terraform.io/app/settings/tokens?source=terraform-login

-
Generate a token using your browser, and copy-paste it into this prompt.

Terraform will store the token in plain text in the following file
for use by subsequent commands:
  /Users/packt/.terraform.d/credentials.tfrc.json

Token for app.terraform.io:
  Enter a value: ?
```

Figure 9.7: HCP Terraform login via CLI waiting for API token

Welcome to HCP Terraform!

Documentation: terraform.io/docs/cloud

New to HCP Terraform? Follow these steps to instantly apply an example configuration:

```
$ git clone https://github.com/hashicorp/tfc-getting-started.git
$ cd tfc-getting-started
$ scripts/setup.sh
```

Figure 9.8: Successful login to HCP Terraform via the CLI

The screenshot shows the HCP Terraform workspace details page for the 'getting-started' workspace. A blue arrow labeled 'Organization' points from the left sidebar to the workspace title. Another blue arrow points from the workspace title to the 'Latest Run' section.

Workspace Overview:

- ID: ws-ar26KTHdnatLmxCuY
- Add workspace description
- Unlocked
- Resources 5
- Terraform v1.7.4
- Updated an hour ago

Latest Run:

- Triggered via CLI
- triggered a run an hour ago via CLI
- Policy checks: Add
- Estimated cost change: Less than a minute
- Plan & apply duration: Less than a minute
- Resources changed: +5 -0
- Status: Applied
- See details

Resources:

NAME	PROVIDER	TYPE	MODULE	CREATED
prod_db	hashicorp/fakew...	fakewebservi...	root	Feb 29 2024

Metrics (last 1 run):

- Average plan duration: < 1 min
- Average apply duration: < 1 min
- Total failed runs: 0
- Policy check failures: 0

Tags (0):

Add a tag

Tags have not been added to this workspace.

Figure 9.9: CLI-driven remote execution mode completion details

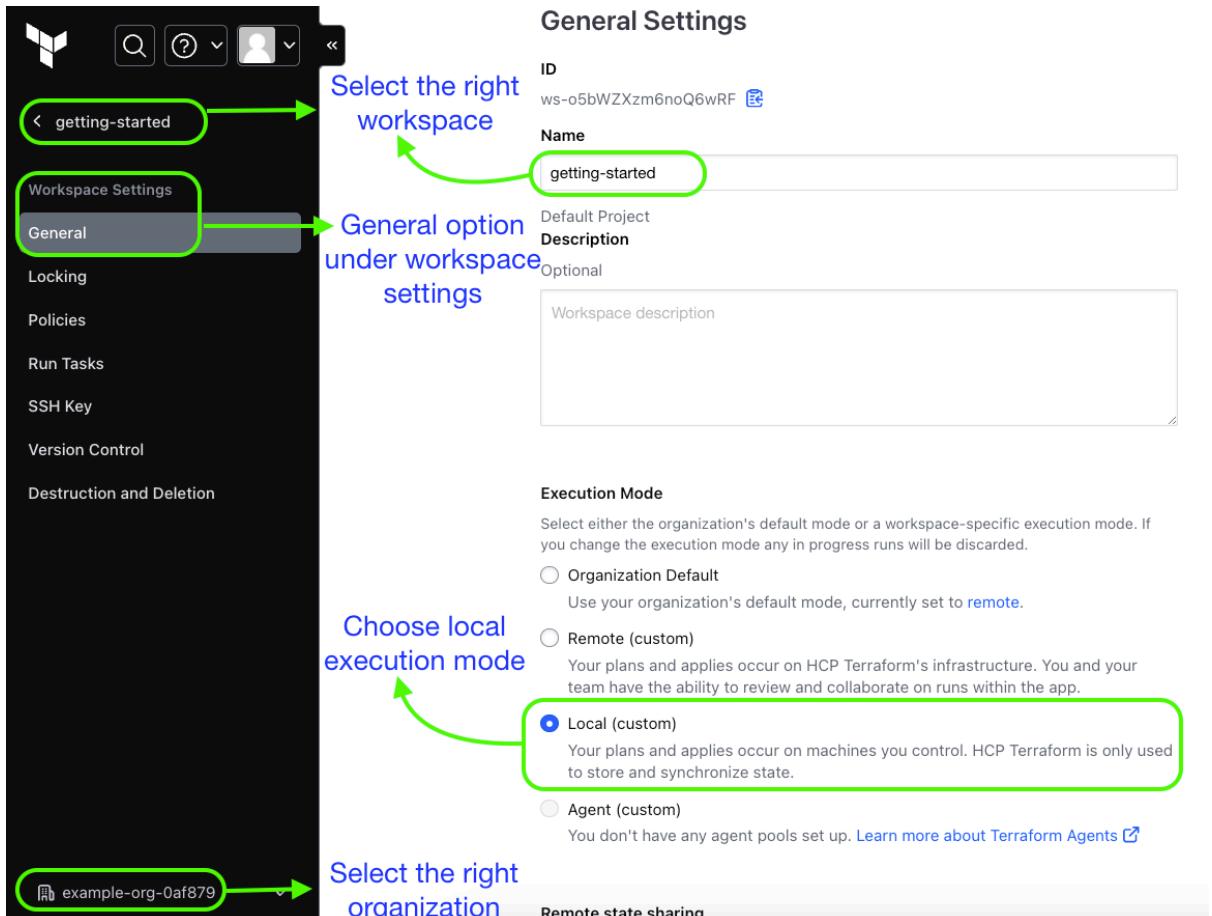


Figure 9.10: CLI-driven local execution mode configuration

```
~/tfc-getting-started $ls
LICENSE          README.md      backend.tf      main.tf        provider.tf    scripts
~/tfc-getting-started $cat main.tf
resource "fakewebservices_vpc" "primary_vpc" {
  name      = "Primary VPC"
  cidr_block = "10.0.0.0/8"      cidr_block is changed to 10.0.0.0/8
}

resource "fakewebservices_server" "servers" {
  count = 10                  Number of servers changed to 10

  name = "Server ${count.index + 1}"
  type = "t2.micro"
  vpc  = fakewebservices_vpc.primary_vpc.name
}

resource "fakewebservices_load_balancer" "primary_lb" {
  name      = "Primary Load Balancer"
  servers  = fakewebservices_server.servers[*].name
}

resource "fakewebservices_database" "prod_db" {
  name      = "Production DB"
  size      = 256
}
```

Figure 9.11: CLI-driven local execution mode changes

Plan: 8 to add, 2 to change, 0 to destroy.

Figure 9.12: terraform plan output for local execution mode

The screenshot shows the Terraform Cloud interface for a workspace named 'getting-started'. The left sidebar has a dark theme with options like 'Workspaces', 'Overview', 'States', and 'Settings'. The main area is titled 'getting-started' with an ID of 'ws-0fc68yyay9flvx4'. It shows an 'Unlocked' status, 13 resources, Terraform v1.7.4, and an update 28 minutes ago. A 'Lock' button is in the top right. The 'Latest Run' section is highlighted, showing it was triggered via CLI 16 hours ago. It's marked as 'Planned and finished'. Below this, there are sections for 'Metrics' (last 1 run) and 'Tags (0)'. The 'Metrics' section includes average plan and apply durations, total failed runs, and policy check failures. The 'Tags' section has an 'Add a tag' button and a note that tags have not been added.

Figure 9.13: CLI-driven local execution mode completion details

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner * Repository name *

/

terraform-vcs-workflow is available.

Great repository names are short and memorable. Need inspiration? How about [solid-guide](#) ?

Description (optional)

This repository has the Terraform configuration files to test VCS workflow

 Public
Anyone on the internet can see this repository. You choose who can commit.

 Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: [Terraform](#) ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: [Apache License 2.0](#) ▾

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

 You are creating a public repository in your personal account.

Create repository

Figure 9.14: Creating a public repository in GitHub

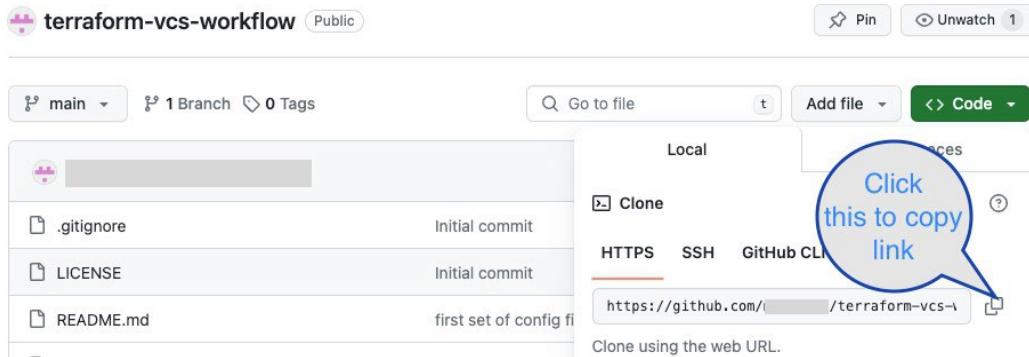


Figure 9.15: Copying the HTTPS link from the GitHub repository

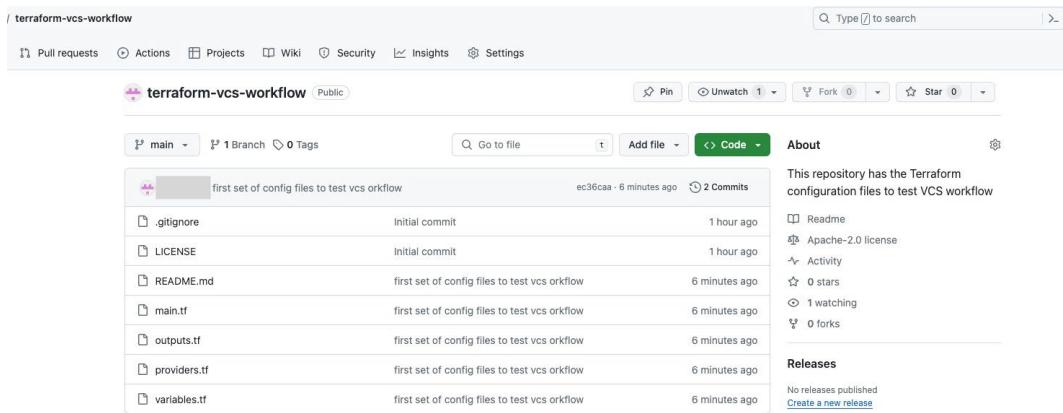


Figure 9.16: Exercise files populated in the GitHub repository

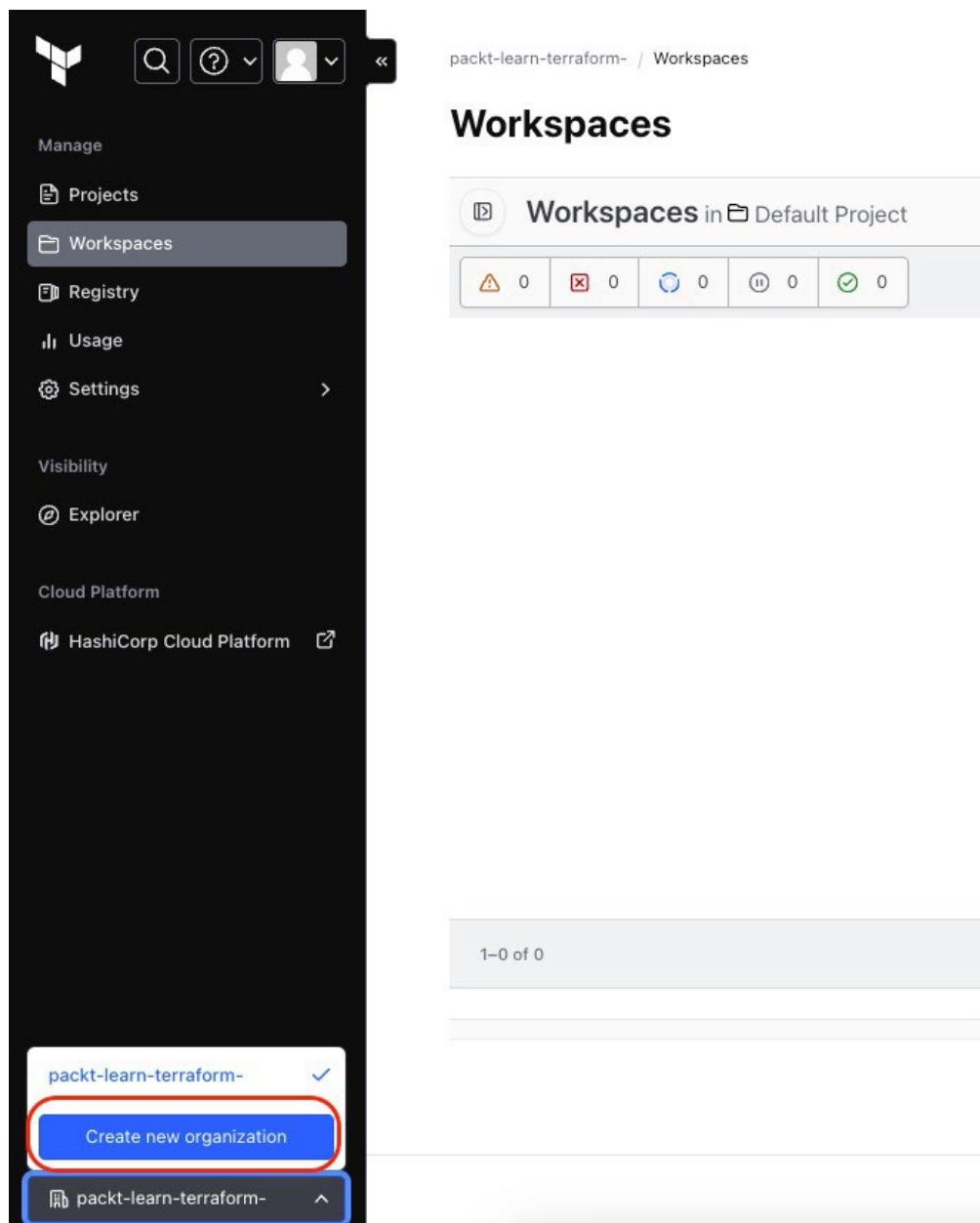


Figure 9.17: Creating a new organization in HCP Terraform

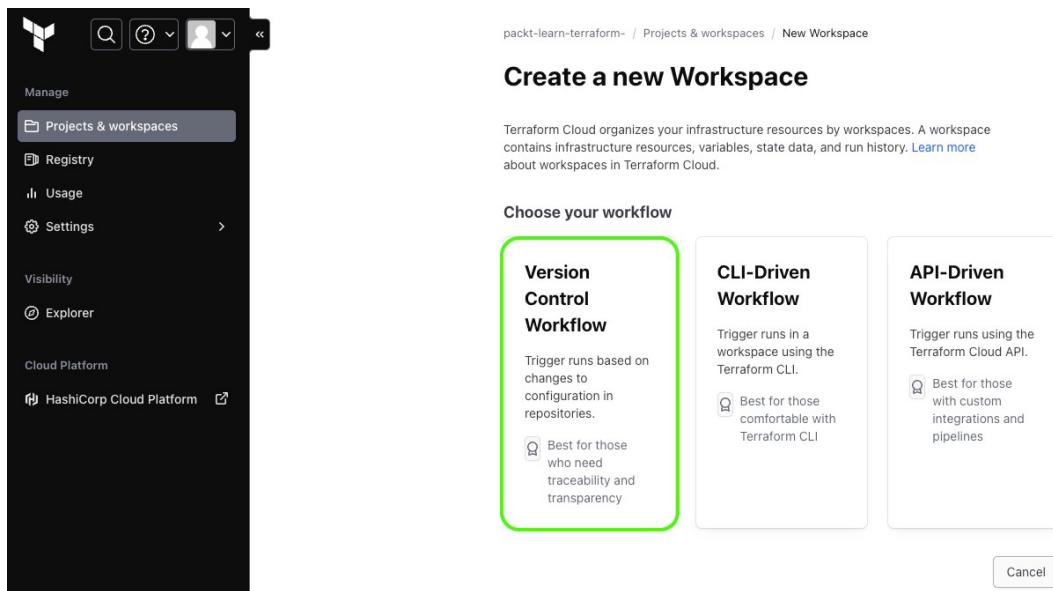


Figure 9.18: Creating a new workspace with Version Control Workflow

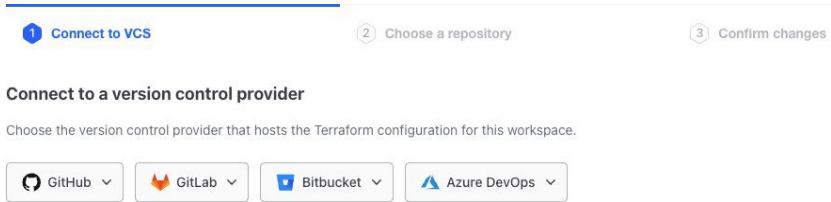


Figure 9.19: Choosing the version control provider

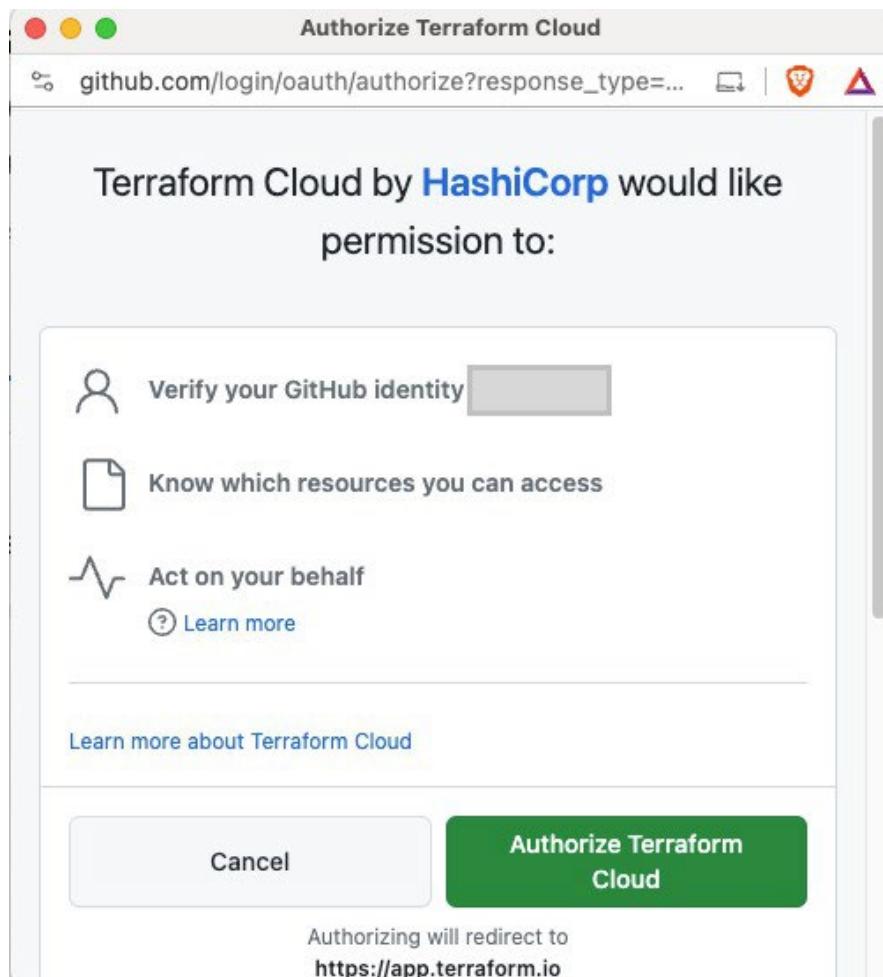


Figure 9.20: Authorization window in GitHub

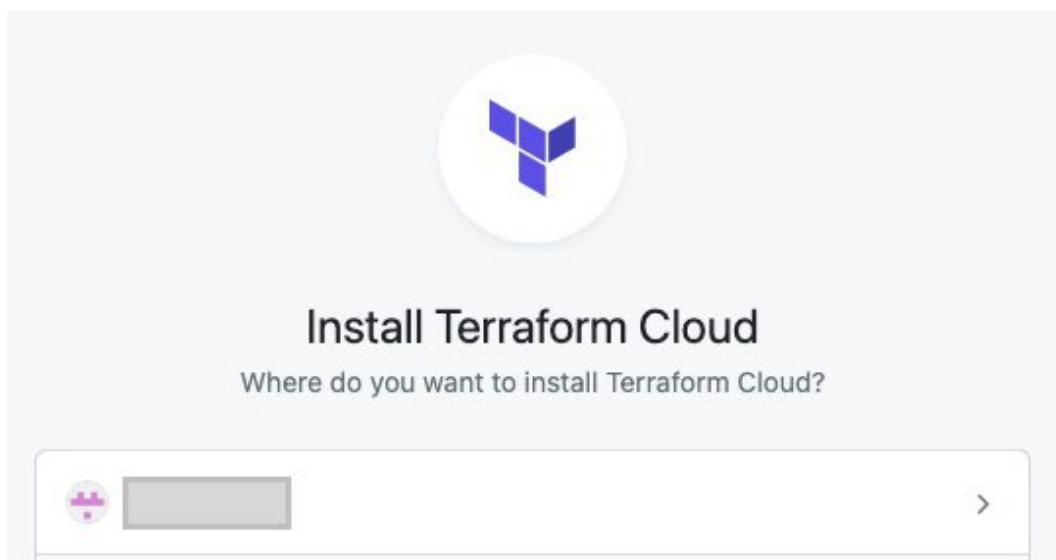


Figure 9.21: Install Terraform Cloud/HCP Terraform in GitHub

Workspace created!

[Go to workspace overview](#)

Workspace ✓ [terraform-vcs-workflow](#)

Next step: Configure Terraform variables

No variables found

Your configuration does not contain input variable definitions that need values entered.

[Continue to workspace overview →](#)

Start your first plan

After you configure any required input variables, start your first plan.

[Start new plan](#)

Figure 9.22: Successful creation of a workspace

Terraform uses all Terraform [variables](#) and Environment [variables](#) for all plans and applies in this workspace. Workspaces using Terraform 0.10.0 or later can also load default values from any `*.auto.tfvars` files in the configuration. You may want to use the Terraform Cloud Provider or the variables API to add multiple variables at once.

Sensitive variables

Sensitive [variables](#) are never shown in the UI or API, and can't be edited. They may appear in Terraform logs if your configuration is designed to output them. To change a sensitive variable, delete and replace it.

Workspace variables (2)

Variables defined within a workspace always overwrite variables from variable sets that have the same type and the same key. Learn more about variable set precedence [»](#).

Key	Value	Category
AWS_ACCESS_KEY_ID AWS Access key	AKIA57I...	terraform
AWS_SECRET_ACCESS_KEY AWS Secret key SENSITIVE	Sensitive - write only	terraform

[+ Add variable](#)

Figure 9.23: Adding AWS credentials as variables

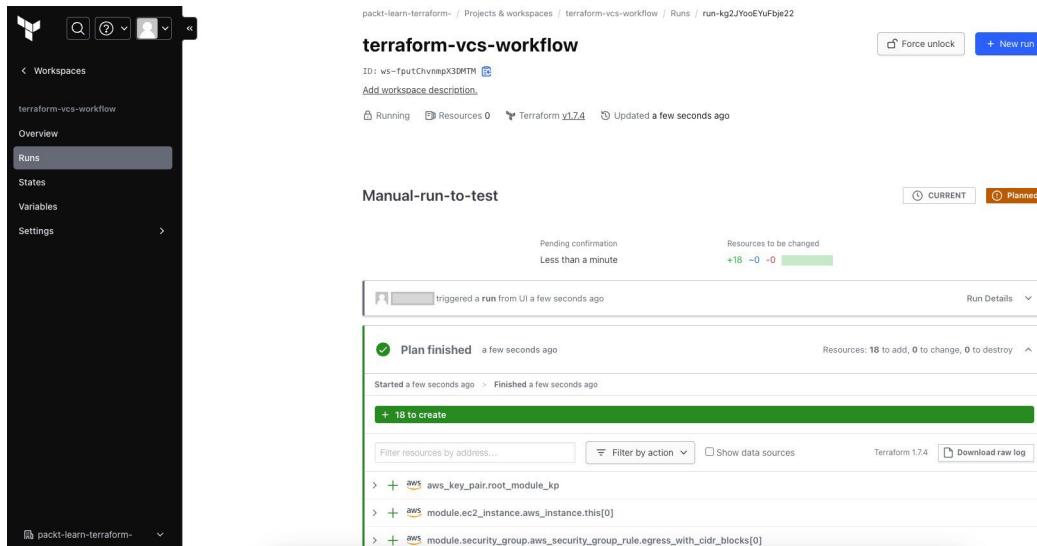


Figure 9.24: terraform plan output for VCS trigger

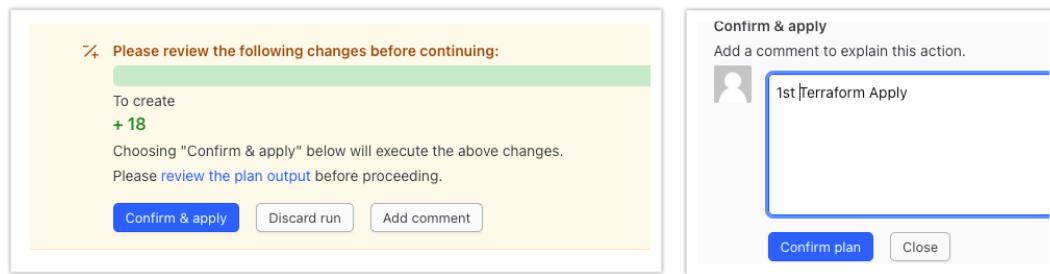


Figure 9.25: Confirm the Terraform plan to apply

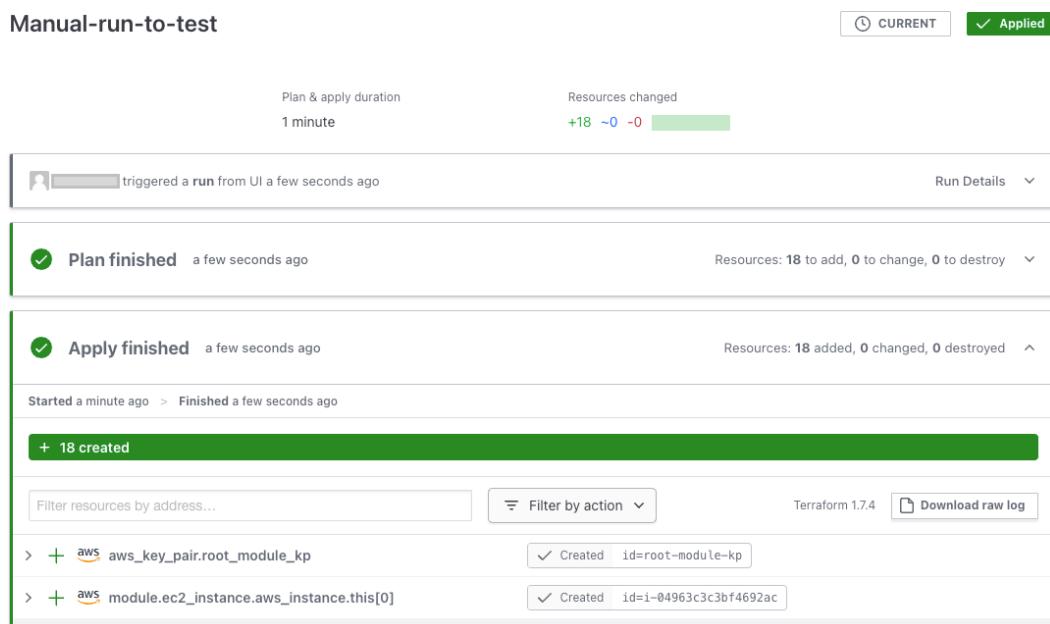


Figure 9.26: Creation of resources after a manual run

```
resource "aws_key_pair" "root-module-kp" {
    key_name    = "vcs-test-kp"
```

Figure 9.27: Changing the keypair name

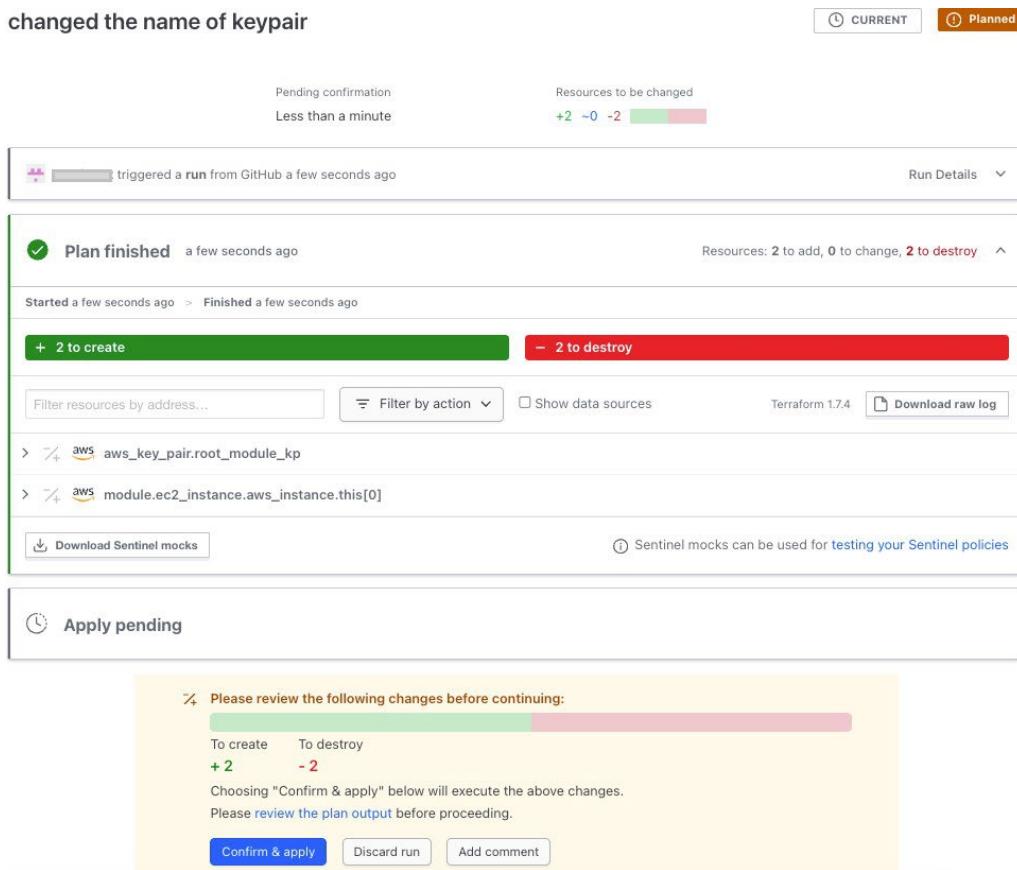


Figure 9.28: terraform plan output for the keypair change in the VCS repository

packt-learn-terraform- / Settings / Cost Estimation

Cost Estimation

Enable Cost Estimation for all workspaces

When possible, display an estimated monthly cost for resources provisioned.

[Update settings](#)

Figure 9.29: Enabling the Cost Estimation feature at the organization level

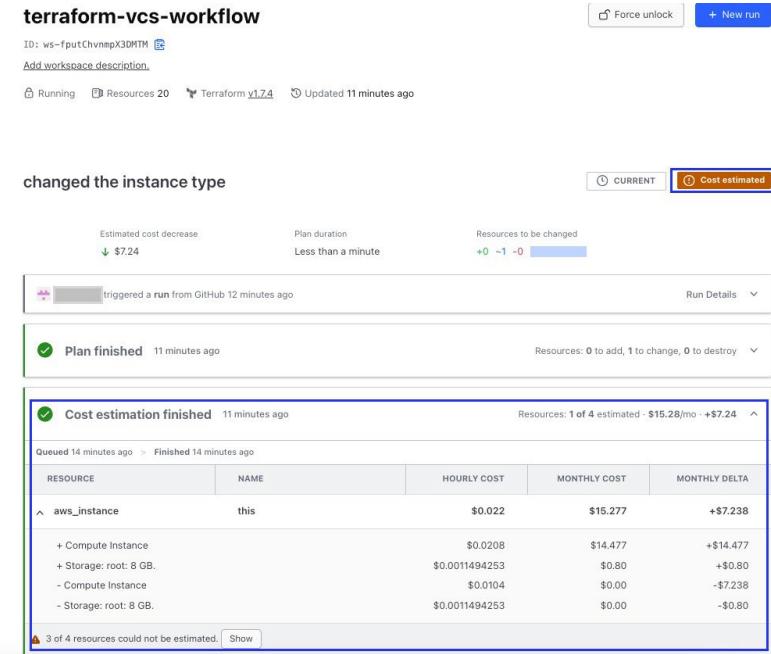


Figure 9.30: terraform plan showing the estimated cost

In this chapter, you have learned about the various Terraform editions at your disposal after you decide to go with Terraform for infrastructure management. By now, you should have a good understanding of the shortcomings of Terraform Community Edition, when to move to HCP Terraform/Enterprise, and the additional features of these editions. You also learned how to sign up for HCP Terraform and looked at a few exercises that helped you understand how the workflows work and what the local and remote execution modes are.

Policy as code is one of the key features that is used to enforce certain policies around costing, best practices, and compliance. You learned how this can be implemented using Sentinel. Toward the end of the chapter, you looked at the unique features of Terraform Enterprise.

In the next chapter, you will look at some miscellaneous topics that have not been covered so far.

Chapter Review Questions

The HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda

Select Quiz

Quiz 1 START SHOW QUIZ DETAILS

Figure 9.32 - Chapter Review Questions for Chapter 9

Chapter 10: Miscellaneous Topics

```
Error: Invalid value for variable

on input-validations.tf line 5:
5: variable "app_id" {
  |
  var.app_id is "app-1234"

The app_id value must include the prefix "app-" and should be 9 characters long.

This was checked by the validation rule at input-validations.tf:9,3-13.
```

Figure 10.1 – Error message returned by Terraform

```
Plan: 1 to add, 0 to change, 0 to destroy.

Warning: Check block assertion failed

on check-assertions.tf line 7, in check "destroy_check":
7:     condition = aws_s3_bucket.app_bucket.force_destroy == true
      |
      aws_s3_bucket.app_bucket.force_destroy is false

The S3 bucket created with the force_destroy parameter as false
```

Figure 10.2 – Error message from the check {} block

```
aws_s3_bucket.app_bucket: Creating...
aws_s3_bucket.app_bucket: Creation complete after 6s [id=terraform-1234567890]
[...]
Warning: Check block assertion failed

on check-assertions.tf line 7, in check "destroy_check":
7:     condition = aws_s3_bucket.app_bucket.force_destroy == true
      |
      aws_s3_bucket.app_bucket.force_destroy is false

The S3 bucket created with the force_destroy parameter as false

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Figure 10.3 – Output from terraform apply

```
aces-example> terraform workspace list
  default
* dev
```

Figure 10.4 – terraform workspace list command output

The screenshot shows a dark-themed web application interface. At the top left is the 'Practice Resources' logo. On the right are a bell icon and a 'SHARE FEEDBACK' button. Below the header, the navigation bar shows 'DASHBOARD > CHAPTER 10'. A section titled 'Miscellaneous Topics' has a 'Summary' link. To the right, a large white box contains text about the focus of the chapter and its importance. To the right of this is a dark sidebar with a title 'Chapter Review Questions', the author information 'The HashiCorp Terraform Associate (003) Exam Guide by Chandra Mohan Dhanasekaran, Manjunath H Gowda', a 'Select Quiz' dropdown, and a 'Quiz 1' section with a 'START' button.

DASHBOARD > CHAPTER 10

Miscellaneous Topics

Summary

In this chapter, the focus was more on topics that are not vast but are very important to understand when you want to build and deploy complex applications using Terraform. Provisioners are very helpful if you know how to use them effectively and handle exceptions that can occur. The usage of sensitive data in Terraform needs to be planned well ahead of the project schedule and external secret engines should be utilized whenever possible.

Chapter Review Questions

The HashiCorp Terraform Associate (003) Exam Guide
by Chandra Mohan Dhanasekaran, Manjunath H Gowda

Select Quiz

Quiz 1

[SHOW QUIZ DETAILS](#) ▾

[START](#)

Figure 10.6 – Chapter Review Questions for Chapter 10

Chapter 11: Accessing the Online Practice Resources



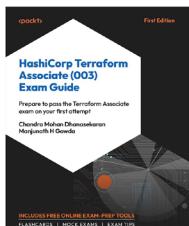
Figure 11.2 – Unlock page for the online practice resources



Figure 11.3 – Enter your unique sign-up code to unlock the resources

PACKT PRACTICE RESOURCES

You've just unlocked the free online content that came with your book.



HashiCorp Terraform Associate (003) Exam Guide

 Book ISBN: 9781804618844

Chandra Mohan Dhanasekaran • Manjunath H Gowda • May 2024 • 400 pages

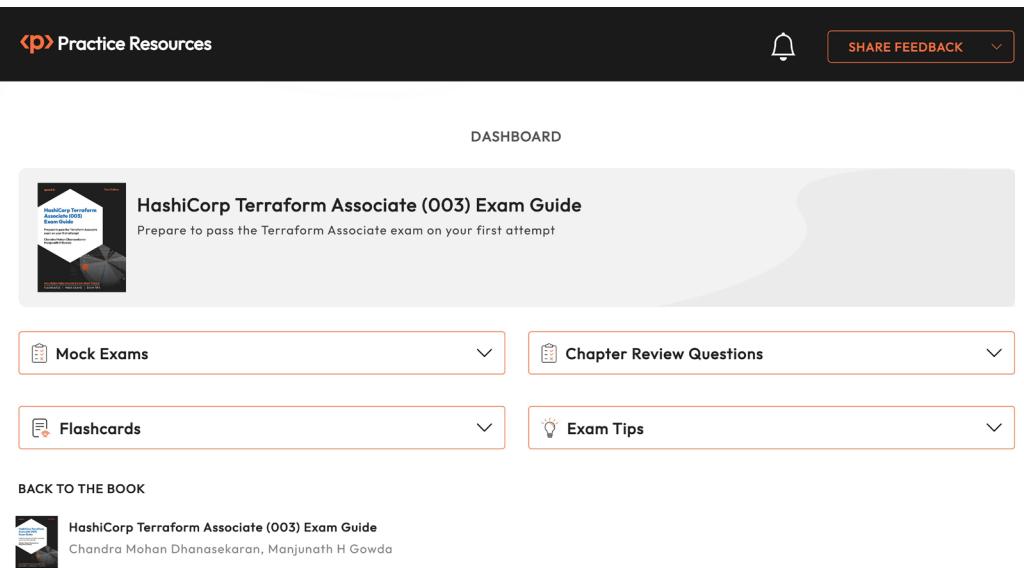
 **Unlock Successful**

Click the following link to access your practice resources at any time.

Pro Tip: You can switch seamlessly between the ebook version of the book and the practice resources. You'll find the ebook version of this title in your [Owned Content](#)

[OPEN PRACTICE RESOURCES](#) 

Figure 11.4 – Page that shows up after a successful unlock



The screenshot shows the 'DASHBOARD' section of the practice resources page. At the top, there are navigation links for 'Practice Resources', a bell icon for notifications, and a 'SHARE FEEDBACK' button. Below this, the main content area displays the book cover for 'HashiCorp Terraform Associate (003) Exam Guide' and its description: 'Prepare to pass the Terraform Associate exam on your first attempt'. Underneath the book info, there are four expandable sections: 'Mock Exams', 'Chapter Review Questions', 'Flashcards', and 'Exam Tips'. At the bottom left, there is a link to 'BACK TO THE BOOK' with the book's thumbnail and title.

Figure 11.5 – Dashboard page for HashiCorp Terraform Associate (003) practice resources

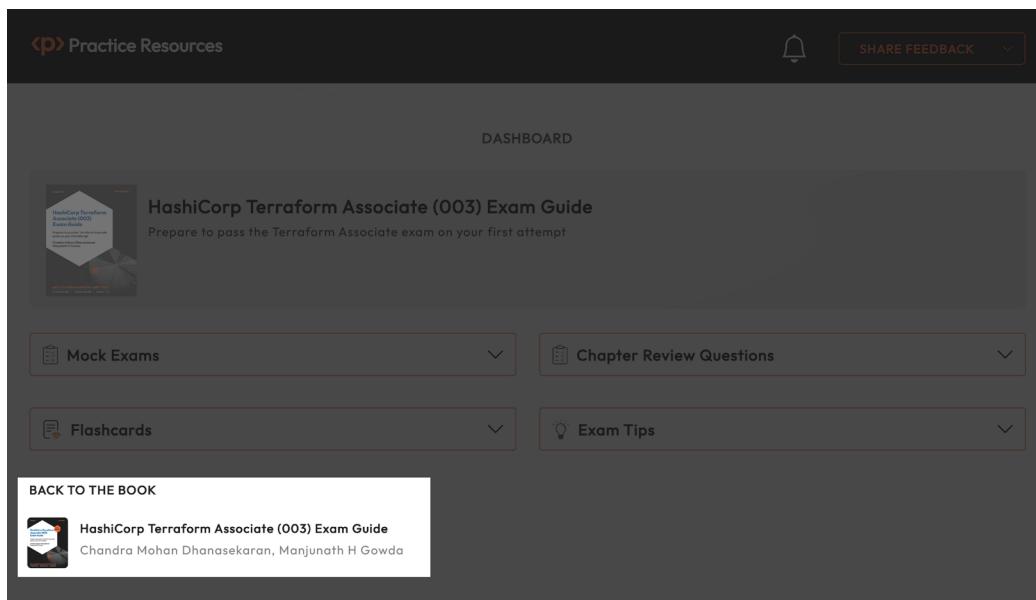


Figure 11.7 – Dashboard page for HashiCorp Terraform Associate (003) practice resources